Leveraging User Experience through Input Style Transformation to Improve Access to Music Search Services

Marina Purgina, Andrey Kuznetsov and Evgeny Pyshkin
St. Petersburg State Polytechnical University
Institute of Computing and Control
Polytechnicheskaya ul. 21, St. Petersburg 194021, Russia
E-mail: mapurgina@gmail.com, andrei.n.kuznetsov@gmail.com, pyshkin@icc.spbstu.ru
http://kspt.ftk.spbstu.ru/info/staff/pyshkin/en

Keywords: information retrieval, music search, mobile application, query transformation, input style transformation

Received: November 23, 2013

We analyze problems of music searching and main tasks the developers face in the domain of music information retrieval. We introduce the architecture of the software and the data model for integrated access to existing music searching web services. We illustrate our approach by developing a mobile accessed software prototype which allows users of Android running touch screen devices accessing several music searchers including Musipedia, Music Ngram Viewer, *and* FolkTuneFinder. *The application supports various styles of music input query. We pay special attention to input style transformation aimed to fit well the requirements of the supported search services.*

Povzetek: Opisana je metoda iskanja glasbenih posnetkov na androidnih napravah.

1 Introduction

A variety of multimedia resources constitutes considerable part of the present-day Web information content. Numerous search services usually provide special features to deal with different types of media such as books, maps, images, audio and video recordings, software, etc. In addition to general-purpose searching systems there are solutions using specialized domain sensitive interfaces. Truly, quality of a search service depends both on the efficiency of algorithms it relies on, and on user interface facilities. Such interfaces may include special syntax forms, user query visualization facilities, interactive assisting tools, components for non-textual query input, interactive and "clickable" concept clouds, and so on [1]. Depending on the searching tasks, specialized user interfaces may support different kinds of input like mathematical equations or chemical changes, geographic maps, XML-based resource descriptions, fragments of software source code, editable graphs, etc.

In text searching such aspects as morphological, synonymic and grammar variations, malapropisms, and spelling errors condition particular difficulties of a searching process. In the music searching domain there are specific complications like tonality changes, omitted or incorrectly played notes or intervals, time and rhythmic errors. Thus, although there are eventual similarities between text and music information retrieval, they differs significantly [2].

Human ability to recognize music is strongly interrelated to listener's experience which may be considered itself to be a product of music intelligent perception [3, 4]. Recently (see [5]) we also analyzed internal models of music representation (with most attention to function-based representation) being the foundation of various algorithms for melody extraction, main voice recognition, authorship attribution, etc. Music processing algorithms use the previous user experience implicitly. As examples, we could cite the *Skyline* melody extraction algorithm [6] based on the empirical principle that the melody is often in the upper voice, or *Melody Lines* algorithms based on the idea of grouping notes with closer pitches [7].

The remaining text of the article is organized as follows. In section 2 we review music searching systems and approaches of the day. We also introduce our experience in the domain of human centric computing and refer to some recent related works. In section 3 we describe music query input styles and analyze possible transformations of music input forms so as to fit the requirements of search services. Section 4 contains the description of the developed Android application architecture. We show how it works and make an attempt to analyze the searching output from the point of a musicologist.

This paper is based on M. Purgina, A. Kuznetsov and E. Pyshkin *An Approach for Developing a Mobile Accessed Music Search Integration Platform* published in the proceedings of the 3rd International Workshop on Advances in Semantic Information Retrieval (part of the FedCSIS'2013 conference).

2 Background and related works

In general, we are able to search music either by metadata description, or by music content. Searching by metadata seems to be very similar to textual searching. Metadata is not necessarily to be restricted by bibliographical data like author, title, artist, conductor, editor, date of publication, etc. They may also include information about performance itself like time signature, tempo, musical instruments, tonality, lyrics and so on. In some situations searching by metadata and searching by content are not so different. Let us consider "A Dictionary Of Musical Themes" [8] which includes short snippets (transposed to C-dur tonality) of musical themes of a composition. These snippets can be used to seek unknown composition by its theme. On one hand, these themes extracted from the original composition constitute metadata, on the other hand, they enable searching a composition by its content. In fact, user can use an ordinary text-based search engine to retrieve compositions represented in such a dictionary. Nowadays exactly the same technique are being used in indexing algorithms and fingerprint algorithms with only few differences: a) metadata are extracted automatically, and b) being sort of pure mathematical abstractions (e.g. hashcodes, fingerprint vectors, etc.) metadata may have no sense for humans.

Since the time when the first dictionaries of musical themes were created, the world dramatically changed. People developed new multimedia carriers requiring more complicated search scenarios:

1. Searching music information by existing audio fragment considered as an input.
2. Searching compositions by human remembrance represented in a form of singed, hummed, tapped or anyhow else defined melody or rhythm fragment.
3. Searching music by lyrics.
4. Searching music by bibliographical data (e.g. title, author etc.).
5. Searching music by keywords (e.g. "scary Haloween music").

Searching by given audio fragments is supported by many specialized search engines like *Audiotag*, *Tunatic* or *Shazam*[9]. As a rule, it is implemented on the basis of so called audio fingerprinting technique. The idea of such an approach is to convert an audio fragment of fixed length to a low-dimensional vector by extracting certain spectral features from the input signal. Then this vector (being a kind of audio spectral fingerprint) is compared to fingerprints stored in some database [10, 11].

In the case of *Searching by human remembrance* scenario we can distinguish two situations. In the first one the search engine deals a monophonic user query representing a main voice, a rhythm, a melodic or rhythmic contour. In the second case the user query represents polyphonic music fragment (e.g. while searching by note score). Errors in user queries condition the main problems of *Searching by human remembrance*. Thus, music fragments comparison algorithms have to be robust in regards to the most popular user errors like expansion, compression, omission or repetition [12, 13]. The another complication is that it is impossible to search directly within the audio resources' binary contents since we usually have no exact faithful audio fragment[2].

Searching by lyrics is not fully automated. An end user is able to use general purpose text search machine (like google, yahoo or yandex) for this task, but text based tools search in existing textual data which is usually published either by author or by music lovers. In theory it's possible to use speech recognition engine for the purpose of lyrics extraction [14], but in practice the recognition quality is not good enough for automatic lyrics transcription. We experimented with Google Voice service and it showed good results for lyrics recognition if a user is singing in silent environment with no background music (it successfully recognized 17 of 20 songs). But if we try to play a broadcasted recording of popular artists, the Google Voice simply ignored the input just like it was nothing sang at all. Indeed, the Google Voice was designed as a speech recognition service, not lyrics transcription service. The problem of automatic recognition of lyrics in singing exists, but this topic is actually out of scope of this research.

The first three cited scenarios are related to the so called *cover song identification* task. However the final goal of such a kind of searching process is not always a song itself (which may be an object of copyright restrictions). The user may be satisfied with obtaining music bibliographical metadata that can be used to look for the song in an online music store. It is exactly what the fourth scenario *Searching music by bibliographical data* means.

Finally, *Searching by keywords* is often implemented through tags annotations. In this case a search query is being parsed to find keywords that could be considered to be tags. Then these tags (i.e. words from a predefined vocabulary of genres, moods, instruments etc.) are used for searching through an annotated database. For example, such technique is used by Last.fm or Pandora online radios.

The search scenarios we explained here cover only the most common tasks. Besides such kinds of usual tasks, there are many other music information retrieval problems including searching compositions by their emotional properties (this task appears in recommendation systems), identifying exact particular performance instead of cover song retrieval, locating a position inside a song (used in score following, for instance), and many others[3].

Similar to other kinds of IR systems, a MIR system usually contains frontend and backend components. In this paper we pay attention mostly to a frontend part communicating with existing searchers, considering a backend system as an *Application Program Interface* (API).

[2]For the user provided sequence "A4 B4 C4" as an input it may happened that a melody that the user is actually looking for does not contain such notes at all.

[3]The overview of the most popular MIR tasks together with descriptions of the recent algorithms can be found at Music Information Retrieval Evaluation eXchange (MIREX) home page (see `http://www.music-ir.org/mirex`).

The extensive description of input styles used by music search engines may be found in [15]. Presently there are many searching web services like *Midomi*, *Musipedia*, *Ritmoteka*, *Songtapper*, *Music Ngram Viewer*, and *FolkTuneFinder* where customers use one of several possible styles to input a music query.

Table 1 represents possible ways to access different music web search services and pays attention to the following facilities:

- **Voice** Using voice recorded from microphone
- **Rhythm** Using tapping/clicking with keyboard, mouse or other input device
- **Tags** Support for keywords or tags
- **Exmpl** Using uploaded audio fragment as an example
- **Lyr** Searching by lyrics
- **Notes** Music score or pitch notation
- **VKB** Virtual keyboard generating note sequence with rhythm
- **URD** Parsons code
- **API** External API (SOAP, REST, etc.)

Nowadays many services are accessible via browsers since they support Web interface features. Another important issue is the possibility to access some services from inside the software applications by using open protocols. It gives the way to create tools which allow users not to be limited by only one service at a time.

Particularly, *Musipedia* service uses SOAP protocol described in [16]. *FolkTuneFinder* and *Music Ngram Viewer* (both are also used in our work as target search services) are based on the REST architectural style and their responses are wrapped in JSON format. Detailed description of the API usage rules and examples for *Music Ngram Viewer* service may be found in [17].

With respect to music inputs styles, existing tools support the following opportunities to define a music fragment:

- To sing or to hum the theme and to transfer the recording to the music search engine.
- To write notes by using one of known music notations directly (e.g. music score, Helmholtz or American pitch notation, MIDI notation, etc.).
- To tap the rhythm.
- To play the melody with the use of a virtual keyboard.
- To use MIDI-compatible instrument or it's software model.
- To enter Parsons code, or to set the melody contour by using "U, R, D" instructions [4] as shown in Figure 1.
- To define keywords or enter text query.
- To record a piece of original composition (e.g. record a composition played on a radio with use of microphone) and to transfer the recording to the music search engine.

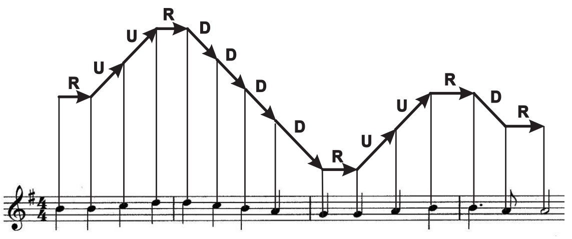


Figure 1: Beethoven's "Ode to Joe" fragment represented in Parsons code.

3 Music query input styles

As shown in the above section, we may define the music query by using different input styles. For a searching framework, the important issue is not only featuring different input interfaces but transforming one query form to the another depending on search service availability and it's communication schema.

Different input styles are useful since the user music qualification differs. Melody definition by using a virtual or real keyboard is one of the most exact ways to represent the query, since it accumulates most melody components. However it is not common that users are skilled enough to use the piano keyboard as well as to write adequate note score.

Contrariwise, tapping a rhythm seems to be relatively simple way to define music searching query. The problem is that the number of possible rhythm patterns is evidently less than the number of compositions. It means that even if we succeed to tap the rhythm correctly, we may apparently have a list of thousands titles in return [5].

If a user didn't record a fragment while the composition was playing, then the only choice is to sing a melody by voice. Recording a voice (so called query-by-humming or query-by-singing) requires both user's singing skills and support for such a facility from the search system. It is important to note that query-by-example and query-by-humming are quite different tasks of MIR[5]).

[4]Each pair of consecutive notes is coded as **U** ("sound goes Up") if the second note is higher than the first note, **R** ("Repeat") if the consecutive pitches are equal, and **D** ("Down") otherwise. Some systems use **S** ("the Same") instead of **R** to designate pitch repetition. Rhythm is completely ignored.

[5]`http://www.music-ir.org/mirex/wiki/2013:Main_Page`

Table 1: Accessing Music Searching Web Services

Name	*Access*								
	Voice	*Notes*	*VKB*	*URD*	*Rhythm*	*Tags*	*API*	*Exmpl*	*Lyr*
Audiotag[a]	–	–	–	–	–	–	–	+	–
Tunatic[b]	–	–	–	–	–	–	–	+	–
Shazam[c]	–	–	–	–	–	+	–	+	–
Midomi[d]	+	–	–	–	–	–	–	–	–
Musipedia[e]	+	+	+	+	+	+	SOAP	–	–
Ritmoteka[f]	–	–	–	–	+	–	–	–	–
Songtapper[g]	–	–	–	–	+	–	–	–	–
Music Ngram Viewer[h]	–	–	+	–	–	+	REST	–	–
FolkTuneFinder[i]	–	–	+	+	+	+	REST	–	–
Google, Yandex, Yahoo!	–	+	–	–	–	+	+	–	+

[a]`http://www.audiotag.info`
[b]`http://www.wildbits.com/tunatic/`
[c]`http://www.shazam.com`
[d]`http://www.midomi.com`
[e]`http://www.musipedia.org`
[f]`http://www.ritmoteka.ru`
[g]`http://www.bored.com/songtapper`
[h]`http://www.peachnote.com`
[i]`http://www.folktunefinder.com`

Mobile devices with touch screens affect strongly usage aspects of music searching interfaces. Such devices make possible simulating many kinds of music instruments, although the virtual piano-style keyboard remains the most popular interface.

3.1 Transformation of input styles

We represent relationships between selected music query input styles in form of an oriented graph. In Figure 2 blue nodes correspond to query representation, red nodes correspond to input methods. *Notes* are used to represent both a query and an input method, and denoted by using grey color. Every transition arc denoted by latin letters (*a* to *r*) shows the possible transformation from one input style to another, namely: a) synthesis & automatic notes transcription; b,c,g) equivalent symbolic transformation; d) pitch estimation; e) pitch sequence with fixed rhythm pattern; f) rhythm with fixed pitch pattern; h) keep only pitch values; i) keep only time intervals; j) calculate pitch intervals[6]; k) calculate inter offset intervals[7]; l) compare pitches; m) compare time intervals; n) pitch sequence with fixed pitch interval pattern; o) rhythm with fixed IOI pattern; p) compare pitch intervals; q) compare IOI; r) onset time estimation.

Since the virtual keyboard based query implicitly includes such note attributes as it's start time, duration and pitch value, there is no much difficulty to transform the keyboard input into the rhythm or pitch notation. The same information (notes) could be extracted from the voice input. Query-by-humming searching machines usually don't need such kind of transformations and use hummed or singed input "as is". However it's still possible to transform singed input into symbolic form in order to create queries for search machines that don't support query-by-humming method. Most recent comparative study of pitch extraction algorithms can be found in [18], and most recent results for multiple fundamental frequency estimation task can be found on MIREX page[8].

Regardless of how we get the notes, we can easily transform them into a rhythm or a pitch notation. This transformation is not lossless. When we transform notes into rhythm we lose information about pitch values, and when we transforming to pitch sequence we lose information about rhythm. Next we can reduce absolute values of pitches and time intervals, and we get sequence of pitch-intervals or sequence of Inter Offset Intervals (IOI). Then we can reduce interval values and get melodic or rhythmic contour. Again this is one-way transformation because we lose an information about the value of an interval and leave only sign of the value encoded with letters 'U', 'R' and 'D'.

Clear that walking through the graph from left to right we reduce the user query, and therefore it seems we couldn't expect better searching results. However such transformations may have sense for at least two reasons:

– We attempt to emphasize the meaning of special

[6]Strictly, this is one way transformation (back transformation produces many transpositions, but as we said before, a comparison algorithm should be robust against transpositions)

[7]Strictly, this is one way transformation (back transformation produces many different tempos, but as we said before, a comparison algorithm should be robust against tempo fluctuation)

[8]`http://www.music-ir.org/mirex/wiki/2013:Multiple_Fundamental_Frequency_Estimation_\%26_Tracking_Results`

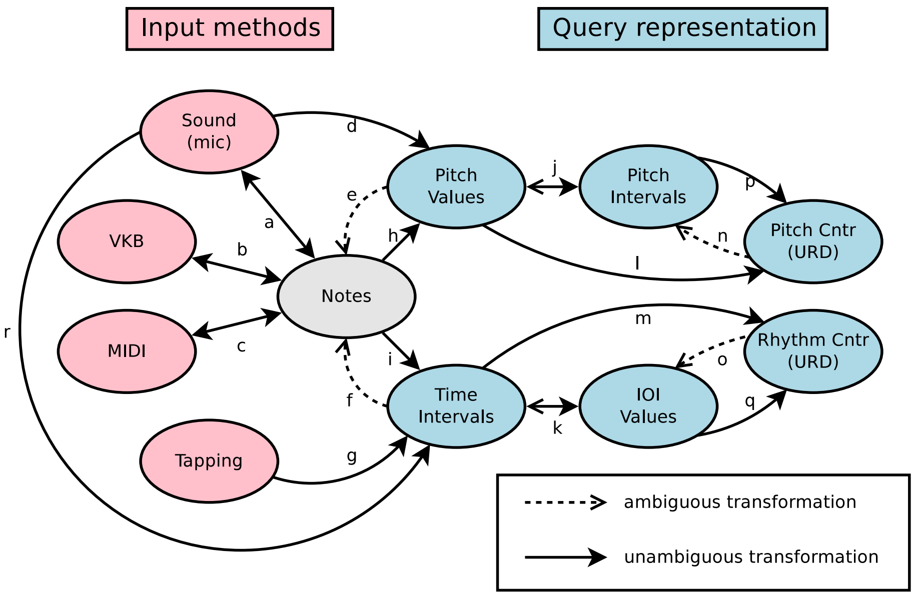


Figure 2: Graph of music query transformations.

melody attributes.

– We would like to try to connect a search service which probably uses quite different music database (e.g. specialized on some music genre[9]) although it supports only restricted input methods (e.g. rhythm or pitch notation).

Regarding to the user interface issues, the ability to move from one input style to another renders possible to switch easily between different searching systems within the framework of one mobile or web application without re-entering the query.

3.2 Models to represent queries

Usually user queries are relatively short (and it is true not only for the case of music [19]), so we use sequence of *Note* objects to represent the searching query.

The attributes of a *Note* object are the following:

– *Note name* according to the American pitch notation

– Its *Octave number*

– Its *Onset time*

– Its *End time*

This representation is equivalent to a subset of MIDI subset of MusicXML format. It means that we can get query in MusicXML format directly from our representation in order to upload it to statistics server. XML based standard is always a good choice for future compatibility. An ability to upload user queries to a server is extremely important feature for the domain of MIR research, because this is the cheapest way to receive information for further investigations [10]. For instance, collected information can be used for modeling user errors.

This representation is equivalent to notes representation so we can easily transform it into the desired form like a sequence of pitches or a rhythm pattern.

4 Introducing mobile application for accessing music search services

Nowadays, people are happy to use their mobile devices to access different search services at any time from any place. They use different types of such devices which may have different input mechanisms like phone keys, *qwerty*-keyboards, touch screens, voice recognition, and so on. The variety of devices running on Android operating system is rapidly increasing during last years, so we decided to use Android platform for our music searching application.

For our implementation we selected some music searchers which may be accessed programmatically, particularly: *Musipedia*, *Music Ngram Viewer*, and *FolkTuneFinder*. For three searching systems we implemented four user query input styles:

– Note score editor supporting one voice definition

– Parsons code

– Rhythm tapping

[9]We turn our attention to the example of such a case in the following section of this paper.

[10]This approach is very similar to a *Game With A Purpose* (GWAP) that is widely used to collect annotations for data. As an example we can cite “Major Miner – music labeling game” for audio data or “Google Image Labeler” for pictures.

– Piano style virtual keyboard with additional representation of American pitch notation

4.1 Application architecture

General application construction ideas are common for various operating platforms. Despite the fact that eventually we developed an application for Android operating system, here we describe common application architecture in platform independent way, but keeping in mind that target device is a mobile device. The Android application can serve as a model for implementing flexible human centric interface which is oriented to present-day style of using hardware and software facilities of various mobile devices.

Figure 4 represents main components of our music searching application.

There are following UI and non UI components shown in Figure 4:

– UI views (boxes with blue background)

– Query transformer

– Search system adapters (one adapter for each search system, boxes with red background)

– Serializers (either JSON or XML)

– Connectors (only one connector supported so far: HTTP)

This is a sort of scalable architecture. We can easily add new query input methods, new search machines or new communication protocols (like FTP or even SMTP if required). The theory of operations is the following. The main view *InputStyleSelection* provides the interface for input style choice. According to the selected input style the respective view (*MelodyContour*, *MusicScore*, *VirtualPiano*, or *RhythmTapper*) opens and provides the corresponding input interface. Depending on the input style the user query could be either a sequence of *Note* objects (Music Score and Virtual Piano produce this output), *Rhythm* (produced by RhythmTapper) or *Melody contour*. Then the application iterates through a collection of available adapters for search engines. Depending on the adapter capabilities the input query may be transformed to the acceptable representation. Then the adapter performs a request to a search service. The request is serialized with a serializer (JSON or XML) and performed through one of available connectors. The response is parsed by the appropriate adapter and added to a list of search results to be displayed by the SearchResults view.

With respect to search services' application interfaces mentioned in section 2, the web information exchange protocol adapters have been implemented.

The SOAP protocol is not recommended for mobile devices since it uses verbose XML format and may be considerably slower in comparison with other middleware technologies. Unfortunately it is the only way to communicate with the *Musipedia* system. In our case, the mentioned SOAP disadvantages shouldn't case concern since the exchange occurs relatively rarely, only when the respective button is pressed by a user, and there is small amount of information being transferred. We use *org.ksoap2* Java package [20] containing classes required for handling SOAP envelopes and literal XML content. To implement interaction with other search services (based on the REST architecture and wrapping their responses in JSON format which is typically more compact in comparison with XML) we use *Google Gson* Java library [21]. It allows converting Java objects into their JSON representation as well as backward converting JSON strings to equivalent Java objects.

4.2 Usage example

The application starts with a welcome screen for the preferred input style selection (Figure 4).

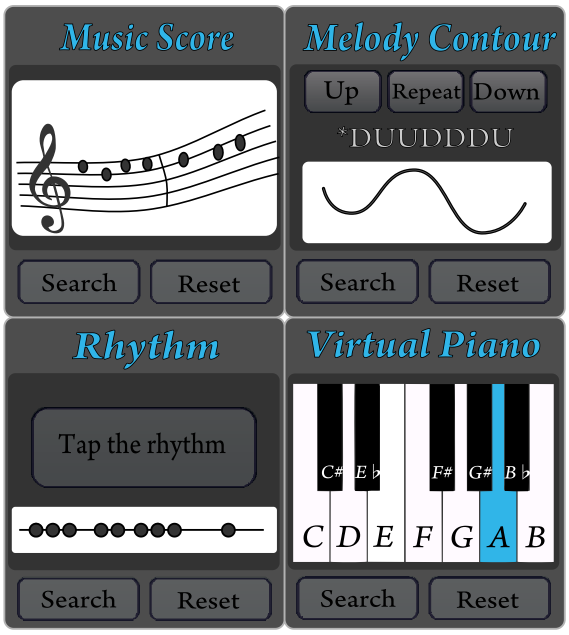


Figure 4: Main activity: input style selection.

Then a user selects an input style. For example if a user selects the virtual keyboard interface, the virtual piano is displayed on the screen. The melody is stored in form of a note sequence with respect to the following related data: a *pitch* represented in the American pitch notation (note name and octave number), *onset time*, *end time*.

Other properties may be computed depending on the requirements of a music searcher. Let us illustrate this by the input represented in form of a simplified timing chart (with respect to the note names rather than sound frequencies). The chart in Figure 5 represents some first notes of the well known Russian folk song "Birch Tree".

For the reason that *Musipedia* searcher requires a sequence of triplets containing an onset time, a MIDI pitch and its duration, the user input shown in Figure 5 is converted to the following query data:

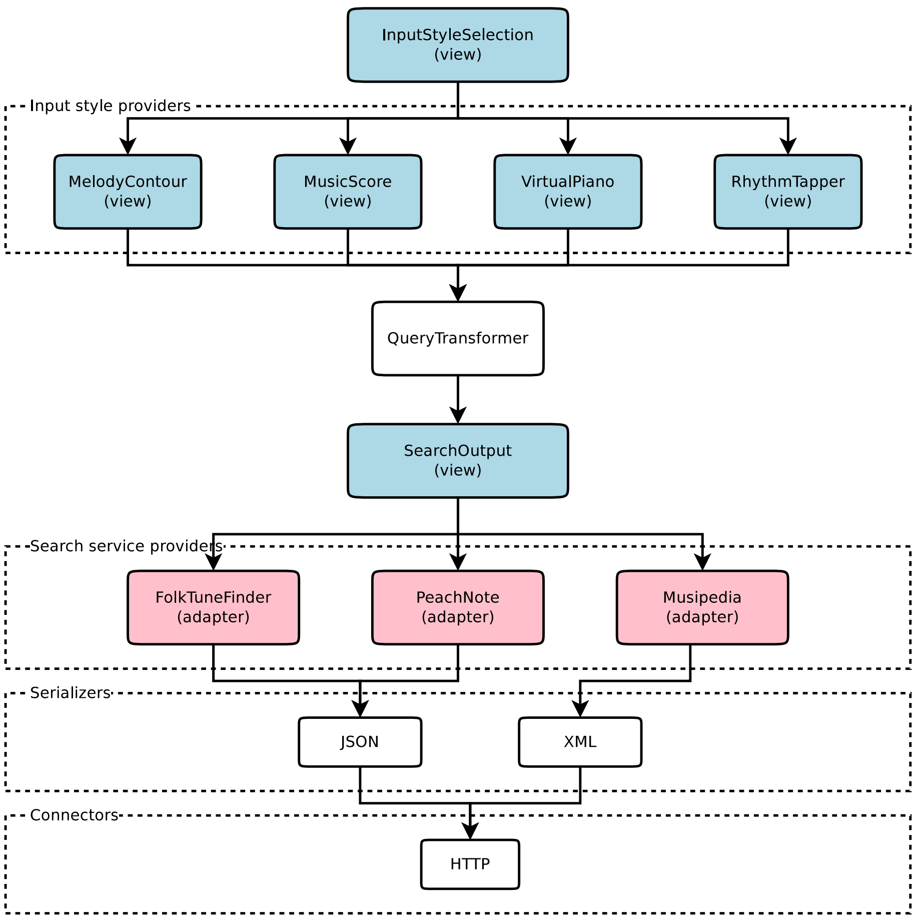


Figure 3: Mobile music searching application architecture.

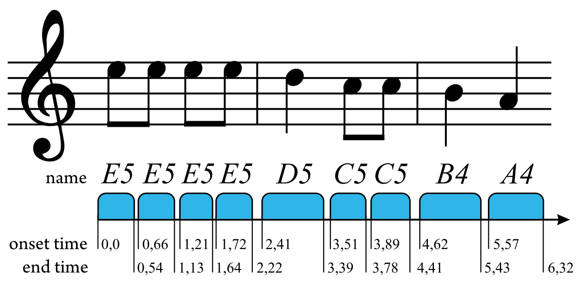
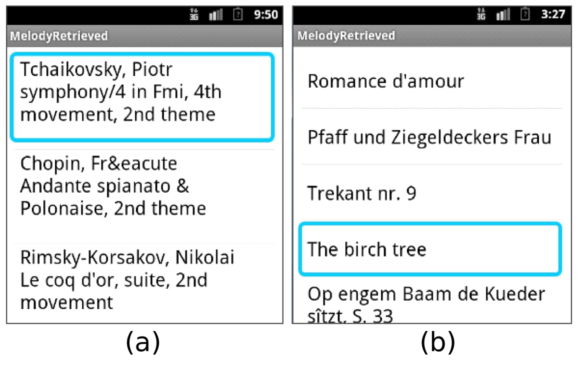


Figure 5: Test melody: note score representation and timing chart.

0.0, 76, 0.54; 0.66, 76, 0.47; 1.21, 76, 1.43; 1.72, 76, 0.50; 2.41, 74, 0.98; 3.57, 72, 0.27; 3.89, 72, 0.52; 4.62, 71, 0.81; 5.57, 69, 0.75;

After this the respective information is included to the SOAP request which is subsequently sent to the *Musipedia* server. As a result, the searching system returns the list of retrieved compositions as shown in Figure 6(a) [11]. We see the confirmation of the known fact that this melody was used by Piotr Tchaikovsky in the 4th movement of his Symphony No.4 in F-moll (compare with the fragment of the symphony note score shown in Figure 7).


9:50
MelodyRetrieved
Tchaikovsky, Piotr symphony/4 in Fmi, 4th movement, 2nd theme
Chopin, Fr&eacute Andante spianato & Polonaise, 2nd theme
Rimsky-Korsakov, Nikolai Le coq d'or, suite, 2nd movement

3:27
MelodyRetrieved
Romance d'amour
Pfaff und Ziegeldeckers Frau
Trekant nr. 9
The birch tree
Op engem Baam de Kueder sîtzt. S. 33

(a) (b)

Figure 6: Results retrieved by *Musipedia* and *FolkTuneFinder* searchers.

Using other searching engines may enhance searching results by taking into account other music genres. Let us take the *FolkTuneFinder* service which requires a sequence of MIDI pitches. Hence the user input is transformed into the sequence of MIDI pitches as follows:

[11]We selected Tchaikovsky's work, but as you can see, the similar theme may also be recognized in some other known compositions.

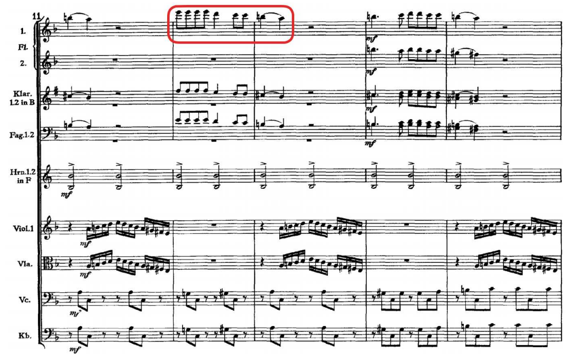


Figure 7: Birch Tree song cited by Tchaikovsky in his 4th symphony.

76, 76, 76, 76, 74, 72, 72, 71, 69

For the case of the melody contour defined with using Parsons code, the user input is the following *"RRRD-DRDD"*. We implemented the interface component which allows constructing the URD-query by pushing buttons *Up*, *Down* and *Repeat* with synchronous demonstration of the respective graphical contour which is being generated automatically[12]. As you see in Figure 6(b) the resulting output also contains the "Birch Tree" among other compositions. Note that since the melody contour is a less exact input method (comparing to direct melody definition), it is normal that we don't have the desired melody in the very first lines.

The example we selected for the illustration shows well one important aspect of music searching process, although in a slightly simplified manner. When we discover the composition corresponding to the given request, we may expect obtaining even more information than simply a desired piece of music. Fast every Russian knows the "Birch Tree" song since the early childhood years. But only those who listen to the classical music discover this theme in one motive of Tchaikovsky's symphony. In contrast to this, western music lovers may listen this motive first just in the Tchaikovsky's work, and after a while recognize it as a citation of the Russian folk song. Isn't it a kind of process similar to a music perception in terms of musicology?

5 Conclusion and future work

In the domain of human-centric computing much attention is paid to the facilitating user interface features in relation with a kind of data being processed. As a special type of information retrieval systems, music retrieval systems demand special ways to interact with users. They include not only traditional text or media based queries but specific forms of user input facilities such as note score representations, virtual or MIDI-compatible instruments, as well as composing queries based on melody humming or rhythm tapping which may contain errors of human interpretation. Such approaches may help to overcome limitations of fingerprinting techniques which require exact or nearly exact audio fragments to proceed with searching in the databases of stored music compositions. In our work we investigated styles of user inputs used in various music search services and applications. We applied transformation rules of query conversion from one input style to another to a software tool communicating with programmatically accessible music search services from mobile devices running on the Android operating system.

In the current implementation we supported only those music queries which are representable in symbolic form (e.g. note score, pitch notation, note sequences, or contour symbolic description). User interface facilities may be improved if we consider other ways to interact with the user having a touch screen device. It may include, for example, melody contour or rhythm drawing facilities. Even for the search services that we used currently, there are input styles which are still not incorporated into the existing software prototype. We investigate possibilities to support interfaces for melody singing or humming. Actually we faced the problem to pass the audio query to the searching engines via existing data transfer protocols that we are allowed to use.

Another way to extend the interface is to provide additional filters like http://www.folktunefinder.com/search/melody/ does. As described in [22], we can provide additional filters or search criteria to experienced users. If a user can provide an information about time signature, tonality, instruments, or define other metadata, there is no objective to prevent the user from doing that. Probably two problems that might appear here is a) our UI will be overcomplicated and b) target search machine may not support such kind of searching. Anyway there is an area for research, how to provide this capability without strong coupling with any of target music search machines.

Ways to extend the interface may also include a support for connected MIDI-compatible devices and text-based searching facilities aimed to explore music metadata information. The another interesting improvement which could fit well especially mobile equipment interfaces is to support music tagging as described for example in [23]. Hence the key idea is to connect different kinds of search services with rich user input facilities so as to follow better the usage style of modern mobile devices.

[12]We consider to investigate possibility to support a melody contour drawing interface in future implementations.

References

[1] E. Pyshkin and A. Kuznetsov, "Approaches for web search user interfaces," *Journal of Convergence*, vol. 1, no. 1, 2010.

[2] Z. Mazur and K. Wiklak, "Music information retrieval on the internet," in *Advances in Multime-*

dia and Network Information System Technologies. Springer, 2010, pp. 229–243.

[3] B. Snyder, *Music and Memory: An Introduction*. Cambridge, Mass. [u.a.]: MIT Press, 2000.

[4] D. Deutch, "Music perception," *Frontiers in Bioscience*, 2007.

[5] A. Kuznetsov and E. Pyshkin, "Function-based and circuit-based symbolic music representation, or back to Beethoven," in *Proceedings of the 2012 Joint International Conference on Human-Centered Computer Environments*. ACM, 2012, pp. 171–177.

[6] A. L. Uitdenbogerd and J. Zobel, "Manipulation of music for melody matching," in *Proceedings of the sixth ACM international conference on Multimedia*, Bristol, United Kngdm, September 1998.

[7] C. Isikhan and G. Ozcan, "A survey of melody extraction techniques for music information retrieval," in *Proceedings of 4th Conference on Interdisciplinary Musicology (SIM'08)*, Thessaloniki, Greece, July 2008.

[8] H. Barlow and S. Morgenstern, *A dictionary of musical themes*. Crown Publishers, 1948. [Online]. Available: http://books.google.ru/books?id=jZ5HAAAAMAAJ

[9] A. Wang, "The shazam music recognition service," *Communications of the ACM*, vol. 49, no. 8, pp. 44–48, 2006.

[10] W. Hatch, "A quick review of audio fingerprinting," McGill University, Tech. Rep., March 2003. [Online]. Available: http://www.music.mcgill.ca/~wes/docs/finger2.pdf

[11] P. Cano, T. Kalker, E. Batlle, and J. Haitsma, "A review of algorithms for audio fingerprinting," *The Journal of VLSI Signal Processing*, vol. 41, no. 3, pp. 271–284, 2005.

[12] R. J. McNab, L. A. Smith, I. H. Witten, C. L. Henderson, and S. J. Cunningham, "Towards the digital music library: Tune retrieval from acoustic input," 1996.

[13] S. Downie and M. Nelson, "Evaluation of a simple and effective music information retrieval method," in *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '00. New York, NY, USA: ACM, 2000, pp. 73–80. [Online]. Available: http://doi.acm.org/10.1145/345508.345551

[14] A. Mesaros and T. Virtanen, "Automatic recognition of lyrics in singing," *EURASIP J. Audio Speech Music Process.*, vol. 2010, pp. 4:1–4:7, Jan. 2010. [Online]. Available: http://dx.doi.org/10.1155/2010/546047

[15] A. Nanopoulos, D. Rafailidis, M. M. Ruxanda, and Y. Manolopoulos, "Music search engines: Specifications and challenges," *Information Processing & Management*, vol. 45, no. 3, pp. 392–396, 2009.

[16] "Musipedia SOAP interface." [Online]. Available: http://www.musipedia.org/soap_interface.html

[17] "Music ngram viewer API." [Online]. Available: http://www.peachnote.com/api.html

[18] O. Babacan, T. Drugman, N. d'Alessandro, N. Henrich, and T. Dutoit, "A comparative study of pitch extraction algorithms on a large variety of singing sounds," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, 2013, pp. 7815–7819.

[19] V. Klyuev and Y. Haralambous, "A query expansion technique using the ewc semantic relatedness measure," *Informatica: An International Journal of Computing and Informatics*, vol. 35, no. 4, pp. 401–406, 2011.

[20] "Package org.ksoap2." [Online]. Available: http://ksoap2.sourceforge.net/doc/api/org/ksoap2/package-summary.html

[21] I. Singh, J. Leitch, and J. Wilson, "Gson user guide." [Online]. Available: https://sites.google.com/site/gson/gson-user-guide

[22] A. Kuznetsov and E. Pyshkin, "Searching for music: from melodies in mind to the resources on the web," in *Proceedings of the 13th international conference on humans and computers*. University of Aizu Press, 2010, pp. 152–158.

[23] K. Bischoff, C. S. Firan, and R. Paiu, "Deriving music theme annotations from user tags," in *Proceedings of the WWW 2009*, 2009.